Assignment #1

Programming Exercise Objective:

In this assignment, you will demonstrate what you have learned with respect to

- Using structs, pointers and arrays
- Using multiple .c source files, header file and makefile
- Input validation and proper error message

Code A: Statement of the phone book problem:

- Build a small phone book program
- To start with, you will use a list of TEN random phone numbers from Greater Ontario with area codes (613 for Ottawa), (416 and 647 for Toronto), (519 for Windsor) and (905 for Niagra Falls). You do not need user input to build the list.
- You will represent each phone number by 10 digits.
- You will create a list of 5 Areas as mentioned above (e.g., 613, 416..) . You do not need user input to build the list.

Remember what you have learned in Lectures so far?

- North American telephone numbers always follow the following structure:
  3-digit area code + 3-digit central office code + 4-digit subscriber
  For example, the Algonquin College main phone number is “(613) 727-4723” where the area code is “613”, the central office (or exchange) is “727” and the subscriber is “4723”.
- Neither the area code nor the central office code will ever start with the digit “0” or “1”.
- Print your 10-digit number in the format: “**(613) 727-4723**”

You will:

- Represent each phone entry by a unique 7-digit phone number (3-digit central office code + 4-digit subscriber), 3-digit Area code, Last Name and First Name.
- Represent each Area with 3-digit Area code (416) and Area Name (e.g., Toronto). You will create five Areas as mentioned above.

When your program will run it will ask for 5 choices:

1. If you choose option 1 you will have to **enter 3-digit area code +7-digit phone number** to get the details of that student
2. If you choose option 2 you will have to **enter 3-digit area code** to get the details of all the students who have phone numbers with that area code.
3. If you choose option 3 you will have to **enter last name of the student** to get the details of that student.
4. If you choose option 4 you will get the details of all the listed area codes.
5. Enter ‘q’ to quit your program.

Sample output may look like:

```
Choose one of the five following options:
Press [1] to get information based on phone number:
Press [2] to get information based on Area Code:
Press [3] to get information based on Last Name:
Press [4] to print all area-code information:
Press [q] to quit:
1

You pressed: 1
Enter Area Code: 416

Enter 7-digit number: 2345678

Your 10-digit phone number was: 416-2345678
Phone number: 416-2345678 belongs to the Student Last1, First1 and his number is from Toronto
```

Code B: Statement of the phone book problem:

To start with:

- you will take user input and
- create a linked list of 5 Areas as (area code 613 for Ottawa), (416 and 647 for Toronto), (519 for Windsor) and (905 for Niagra Falls).
- Insert node at the end: You will insert each new area node at the end of the link list.
- You will print the list of area names and count number of listed areas in the link list each time you insert a new area node. This will show that you are inserting a new area node at the end and you can traverse through the link list and determine the length of the list.

```
typedef struct Area
        {
        int areaCode;
        char areaName[20];
        struct Area *nextArea;
        }
        Area;
```

- Take user input and create a link list of 10 phone book entry with the existing 5 areas
  - To start with, you will use a list of 10 random phone numbers from Greater Ontario with area codes according to the following struct.

```
typedef struct PhEntry
{
    int areaCode;
    int phoneNumber;
    char lastName[20];
    char firstName[20];
    struct PhEntry *next PhEntry;
} PhEntry;
```

  - Insert node at the beginning: Insert each new PhEntry node at the beginning of the link list.
  - You will print the listed 10 digit phone number and count number of listed PhEntry in the link list each time you insert a new PhEntry node. This will

show that you are inserting a new node at the head and you can traverse through the link list and determine the length of the list.

- You will implement deletion of a selected node from link list for option 4 below.
- You will implement doubly link for the deletion of a selected node from link list for option 5 below.

When your program will run, it will ask for the following choices:

1) If you choose option 1 you will have to **enter 3 digit area code** + area description. Insert each area node at the end of the area link list and print as detailed above.
2) If you choose option 2 you have to **enter 3 digit area code +7 digit phone number** and last name and first name of the person. Insert each PhEntry node at the beginning of the PhEntry link list and print as detailed above.
3) If you choose option 3 you will be able to modify an existing phone book entry based on:
    - **3 digit area code +7 digit phone number** or
    - **last name** of the person
4) If you choose option 4 you will be able to delete an existing phone book entry based on:
    - **3 digit area code +7 digit phone number** or
    - **last name** of the person

5) If you choose option 5, you will be asked to provide
    - an existing area code or
    - an Area Description.
    - If the area code exists in any of person's Phone book entry then you will be informed: this area is use and cannot be deleted. Otherwise, you may delete that.
    - If the area description uses multiple area code (Toronto has more than one area code), you have to inform that and ask for which area code
    - **You have to implement this using doubly link list and**
    - **This a bonus option worth 5 marks.**
6) Enter 'q' to quit your program.

Sample options may look like:

```
Choose one of the following options:
Press [1] to Enter Area information:
Press [2] to Enter PhoneBook Entry:
Press [3] to Modify an existing PhoneBook Entry:
Press [4] to Delete an existing PhoneBook Entry:
Press [5] to Delete an unused Area using doubly link list:
Press [q] to quit:
```

Requirements:

1. Create a folder called algonquinUserID1_algonquinUserID2_A1 (e.g., "mynam00123_yourna45678_A1") that represents the two members of your team. Do all of your work in this folder, and when complete, submit the zipped folder as per the "Lab Instructions" posted on Brightspace.

2. Write a program that will implement ALL the requirements, explicit and implicit, listed in the "Statement of the problem" above.

3. You must distribute your functions in a meaningful manner across **multiple .**c files.

a. The .c files should contain functions that represent sensible groupings of functionality
b. You must define .h files as appropriate
4. Each function must have a header comments that explain what it does, and describe/explain its inputs (if any) and return value (if any)
5. You may NOT use any global variables that are defined in a different file. I.e., you have to pass arrays as function arguments! You will implement link list and use pointers to struct.
6. You must use a „makefile", with the **CC_FLAGS** set to "**-g –ansi –pedantic –Wall –w**"), and **OUT_EXE** set to "**assignment1**"

Marking:

This assignment is out of 40:

- 10 for coding correctness using struct and array (i.e., correct results)
- 10 for appropriate and efficient logic (i.e., no spaghetti code!)
- 05 for sensible distribution of well-contained functions between files and makefile
- 15 for validation, proper error message and clear comments and coding convention (not necessarily K&R, but it should be clean and consistent)

Submission:

- When you are done, submit your program to Brightspace, as per the instructions for each assignment.
- You ***must*** include a makefile, as part of your submission!